REMARKS

Claims 1, 3, 8, 90, 14, 17-21, 26, 27, 40, 49, 52, 53, 56, 58 and 63-70 have been amended. Claims 16, 51 and 65 have been canceled. Therefore, claims 1-15, 17-50, 52-64 and 66-70 are now pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 112, Second Paragraph, Rejection:

The Office Action rejected claims 1, 3, 9, 20, 26, 27, 40, 49, 51, 56 and 58 under 35 U.S.C. § 112, second paragraph, as indefinite. Claims 1, 3, 9, 20, 26, 27, 40, 49, 56 and 58 have been amended to overcome the respective section 112 rejections (as noted above claim 51 has been cancelled). Thus, Applicants respectfully request removal of the 35 U.S.C. § 112 rejection of claims 1, 3, 9, 20, 26, 27, 40, 49, 56 and 58.

Section 102(e) Rejection:

The Office Action rejected claims 1-9, 12, 13, 26-32, 35, 36, 49-53 and 56-60 under 35 U.S.C. § 102(e) as being unpatentable over Swift et al. (U.S. Patent 6,377,691) (hereinafter "Swift"). Applicants respectfully traverse this rejection for at least the reasons presented below.

Regarding claim 1, Swift does not disclose determining an access control model to be used by the second node in controlling access, by the first node, to resources of the second. The Examiner cites column 4, lines 1-25 of Swift and states, "Swift discloses using challenge-response to control access to resources." Applicants respectfully disagree with the Examiner's interpretation of Swift.

Swift discloses a system for using a challenge-response authentication protocol for datagram-based remote procedure calls (Swift, Abstract). However, Swift does not teach that challenge-response authentication is used for controlling access to resource. In

contrast, the challenge-response authentication protocol, like authentication protocols in general, is used to verify the identity of a person, computer, or other entity.

The portion of Swift cited by the Examiner describes the communications between a client and a server in Swifts challenge-response authentication protocol. However, the cited passage makes no mention of determining an access control model to be used by the second node in controlling access to resources of the second node. Instead, the Examiner's cited passage describes how a client sends a request for a remote-procedure-call to a server. In response the server sends a challenge to the client and the client in turn sends a challenge response back to the server. The challenge response includes a unique identifier (supplied by the server in the challenge) encrypted with the client's password. The server uses its own copy of the client's password to encrypt the same unique identifier and compare the encrypted result with the encrypted version of the unique identifier from the client's challenge response. The result of this comparison determines the authenticity of the client. Specifically, Swift states, "[i]f the two versions of the challenge response are identical, the identity of the user of the client computer has been verified." (Swift, column 4, lines 15-17). Swift further teaches that after the client user has been authenticated, the server invokes the requested remote procedure call. Thus, beyond verifying the identity of the user, Swift does not determine any access control model to be used by the second node in controlling access, by the first node, to resources of the second.

Thus, the Examiner's cited passage, as well as the remainder of Swift, fails to disclose determining any such access control model. Instead, as noted above, Swift only teaches authenticating the user of a client computer. Authentication and access control are two different concepts. Authentication verifies the identity of an entity while access control determines who has access to what resources. Swift is concerned with the former, i.e. authentication, and is not concerned with the latter, i.e. access control. Merely authenticating a user before invoking a requested remote procedure call cannot be considered determining an access control model.

In further regard to claim 1, Swift also fails to disclose plugging in an access control context *pluggable* module for the determined access control model on the second node, wherein the access control context module is configured for use in controlling access by the first node to resources of the second node using the access control model. The Examiner cites the Summary of Swift. However, as noted above, Swift does not disclose anything regarding access control and further fails to disclose anything regarding plugging in an access control context pluggable module for the determined access control model on the second node.

Also, as admitted by the Examiner in the § 103 rejection of claim 14, Swift does not teach *pluggable* modules. Claim 1 recites plugging in first and second authentication protocol handler *pluggable* modules as well as plugging in an access control context *pluggable* module.

Thus, for at least the reasons given above, the rejection of claim 1 is not supported by the prior art and removal thereof is respectfully requested. Remarks similar to those above apply to claims 26 and 49.

In further regard to claim 26 and regarding claim 56, Swift fails to disclose wherein the first node authentication information is generated by a pluggable first authentication protocol handler module on the first node for the determined authentication type. The Examiner fails to address this limitation in his rejection of claims 26 and 56. Additionally, both claim 26 and claim 56 include plugging in pluggable modules. Specifically, claim 26 recites (in part) the second node plugging in a second authentication protocol handler *pluggable* module and an access control context *pluggable* module. Claim 56 recites (in part) a server executable to plug in a server-side authentication protocol handler *pluggable* module. The Examiner admits, in the § 103 rejection of claim 14, that Swift fails to teach pluggable modules. Thus, the rejection of claims 26 and 56 is not supported by the cited prior art and its removal is respectfully requested.

Regarding claim 58, Swift does not disclose a server executable within a server system to: determine an access control model to be used by the server for the client application. The Examiner cites column 4, lines 1-25 of Swift and states, "Swift discloses using challenge-response to control access to resources." Applicants respectfully disagree with the Examiner's interpretation of Swift. As noted above regarding claim 1, Swift is not concerned with access control. Please refer to the argument above regarding claim 1 for a more detailed discussion regarding Swifts failure to teach access control.

Thus, the Examiner's cited passage, as well as the remainder of Swift, fails to disclose determining an access control model. Instead, as noted above, Swift only teaches authenticating the user of a client computer. Authentication and access control are two different concepts. Authentication verifies the identity of an entity while access control determines who as access to what resources. Swift is concerned with the former, i.e. authentication, and is not concerned with the latter, i.e. access control. Merely authenticating a user before invoking a requested remote procedure call cannot be considered determining an access control model.

In further regard to claim 58, Swift also fails to disclose plugging in an access control context pluggable module for the determined access control model. The Examiner cites the Summary of Swift. However, as noted above, Swift does not disclose anything regarding access control and further fails to disclose anything regarding plugging in an access control context pluggable module for the determined access control model on the second node. Furthermore, as noted above Swift does not teach anything regarding using pluggable modules.

Thus, the rejection of claim 58 is not supported by the prior art and removal thereof is respectfully requested.

Section 103(a) Rejection:

The Office Action rejected claims 14-21, 24, 25, 37-44, 47, 48 and 63-70 under 35 U.S.C. § 103(a) as being unpatentable over Swift et al. in view of Samar (1996 ACM 0-89791-829-0/96-03).

Regarding claim 14, Swift in view of Samar fails to teach or suggest program instruction executable within a second node to: determine an access control model to be used by the second node; and initialize an access control context module for the determined access control model, wherein the access control context module is configured for use in controlling access to resources of the second node by the first node using the access control model. Neither Swift nor Samar mentions either determining an access control model or initializing an access control context module for the determined access control model.

Swift discloses a system for using a challenge-response authentication protocol for datagram-based remote procedure calls (Swift, Abstract). However, Swift does not teach that challenge-response authentication is used for controlling access to resource. In contrast, the challenge-response authentication protocol, like authentication protocols in general, is used to verify the identity of a person, computer, or other entity. However, Swift is completely silent regarding determining an access control model. Instead, as noted above, Swift only teaches authenticating the user of a client computer. Authentication and access control are two different concepts. Authentication verifies the identity of an entity while access control determines who as access to what resources. Swift is concerned with the former, i.e. authentication, and is not concerned with the latter, i.e. access control. Merely authenticating a user before invoking a requested remote procedure call cannot be considered determining an access control model.

Samar teaches a Pluggable Authentication Module (PAM) framework that provides pluggability for a variety of system-entry services. Samar, like Swift is concerned with authentication but not access control. Nowhere does Samar mention determining an access control model. Samar further fails to teach anything regarding initializing an access control context module for the determined access control model.

Thus, Swift and Samar, either singly or in combination, fail to teach or suggest program instructions executable within a second node to: determine an access control model to be used by the second node; and initialize an access control context module for the determined access control model, wherein the access control context module is configured for use in controlling access to resources of the second node by the first node using the access control model. Instead the combination of Swift and Samar would result in a system that provides only authentication without providing any support for access control.

For at least the reasons presented above, the rejection of claim 14 is not supported by the prior art and removal thereof is respectfully requested. Remarks similar to those above regarding claim 14 also apply to claim 63.

Regarding claim 37, Swift in view of Samar fails to teach or suggest program instruction executable within a second node to: determine an access control model to be used by the second node; and initialize an access control context module for the determined access control model, wherein the access control context module is configured for use in controlling access to resources of the second node by the first node using the access control model. The Examiner fails to address these limitations of claim 37 in the rejection of claim 37. Furthermore, as noted above regarding claim 14, neither Swift nor Samar mentions either determining an access control model or initializing an access control context module for the determined access control model. Please refer to the remarks above regarding claim 14, as they apply to claim 37 with equal force.

Thus, Swift and Samar, either singly or in combination, fail to teach or suggest program instruction executable within a second node to: determine an access control model to be used by the second node; and initialize an access control context module for the determined access control model, wherein the access control context module is configured for use in controlling access to resources of the second node by the first node using the access control model. Instead the combination of Swift and Samar would result

in a system that provides only authentication without providing any support for access control.

For at least the reasons presented above, the rejection of claim 37 is not supported by the prior art and removal thereof is respectfully requested.

The Office Action rejected claims 10, 11, 22, 23, 33, 34, 45, 46, 54, 55, 61 and 62 under 35 U.S.C. § 103(a) as being unpatentable over Swift in view of Samar and in further view of O'Brien (U.S. Patent 6,351,776). Applicants traverse this rejection for at least the reasons given above in regard to the independent claims.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any extension of time (under 37 C.F.R. § 1.136) is necessary to prevent the above referenced application from becoming abandoned, Applicants hereby petition for such extension. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-91900/RCK.

Also enclosed herewith are the following items:

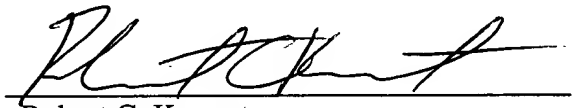
☒ Return Receipt Postcard

☐ Petition for Extension of Time

☐ Notice of Change of Address

☒ Information Disclosure Statement

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: March 10, 2005